RNN-ABC: a new swarm optimization based technique for anomaly detection

Qureshi, Ayyaz-Ul-Haq; Larijani, Hadi; Mtetwa, Nhamoinesu; Javed, Abbas; Ahmad, Jawad

Published in:
Computers

DOI:
10.3390/computers8030059

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

computers

MDPI

Article

RNN-ABC: A New Swarm Optimization Based Technique for Anomaly Detection

Ayyaz-Ul-Haq Qureshi [1,*], Hadi Larijani [1], Nhamoinesu Mtetwa [1] and Abbas Javed [2] and Jawad Ahmad [1]

[1] School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow G40BA, UK
[2] Department of Electrical and Computer Engineering, COMSATS University Islamabad, Lahore Campus 54000, Pakistan
* Correspondence: ayyaz.qureshi@gcu.ac.uk

Received: 15 July 2019; Accepted: 11 August 2019; Published: 14 August 2019

Abstract: The exponential growth of internet communications and increasing dependency of users upon software-based systems for most essential, everyday applications has raised the importance of network security. As attacks are on the rise, cybersecurity should be considered as a prime concern while developing new networks. In the past, numerous solutions have been proposed for intrusion detection; however, many of them are computationally expensive and require high memory resources. In this paper, we propose a new intrusion detection system using a random neural network and an artificial bee colony algorithm (RNN-ABC). The model is trained and tested with the benchmark NSL-KDD data set. Accuracy and other metrics, such as the sensitivity and specificity of the proposed RNN-ABC, are compared with the traditional gradient descent algorithm-based RNN. While the overall accuracy remains at 95.02%, the performance is also estimated in terms of mean of the mean squared error (MMSE), standard deviation of MSE (SDMSE), best mean squared error (BMSE), and worst mean squared error (WMSE) parameters, which further confirms the superiority of the proposed scheme over the traditional methods.

Keywords: intrusion detection; swarm intelligence; artificial bee colony; neural networks; NSL-KDD

1. Introduction

In the current era, dependence upon information systems connected through the Internet has grown rapidly. Such systems enable businesses to increase productivity by extending activities in a smart way. Internet systems maintain high availability, due to which becoming connected is very simple and anyone can start using the internet in a matter of seconds. Such a high availability and ease of access has attracted hackers and given rise to cyber-attacks [1]. At present, the expertise required to attack a networked system is decreasing, while there is also an increasing trend in the sophistication of attacks, thus establishing an inverse relationship. Smart devices and services such as health care systems generate data upon communication and there is a crucial need to secure this useful information from unauthorized access. Many intelligent context-aware access control mechanisms have been developed [2–5] for this purpose. For further assurance, a security application is usually developed to monitor traffic coming into the network but, since personal-use services inside an enterprise have to access the world wide web, these security applications have to let these packets outside as well. Attackers have used this vulnerability and tapped the data packets to manipulate the information and obtain unauthorized access [6,7]. To address this issue, we must develop an intelligent intrusion detection system (IDS) that can learn from the data and ensure controlled access to the end-user system.

An IDS monitors traffic by observing patterns of activities and generating an alarm when the network is under attack from an unauthorized source or sources. The detection may be signature- or anomaly-based [1]. An IDS which compares traffic patterns to an existing set of patterns (i.e., signatures) are known as signature-based or misuse intrusion detection (M'IDS) systems. The M'IDS is better for small-scale enterprises, but high computational load and the inability to detect novel attacks are major disadvantages of such systems. On the contrary, systems which detect attacks from the baseline behavior of the system are known as anomaly-based intrusion detection systems (A'IDS). An A'IDS works by learning from normal behavior and generating an alarm upon intrusion. This process enables it to detect existing as well as novel attacks.

In order to learn from the existing data, several approaches have been presented in the literature regarding the design of IDS using machine learning (ML). Different heuristic algorithms have been used to train ML models which have the ability to perform complex mathematical calculations. Stochastic, iterative, or population-based solutions are utilized to reduce the error in prediction of attacks, where evolutionary algorithm (EA) [8] or swarm intelligence (SI) [9,10] based methods are used. As this paper is an extension of the article [11], we aim to use the SI approach towards the development of an intelligent IDS. In SI, a collective goal is achieved by the interaction of different agents working in the same environment [12].

This paper aims to use artificial bee colony optimization (ABC) to train a machine learning model for the prediction of novel attacks. Artificial bee colony optimization (ABC) is meta-heuristic algorithm which was inspired by food-search behaviour of honey bees in a hive [12]. It uses a population based approach to find better solutions for a given function. It is robust in nature, as the failure of a single bee search does not effect the performance of the whole system. The bees are divided into three groups: Employed bees, scout bees, and onlooker bees. The random search process for the food source, which represents a solution in the search space, is started by a scout bee. Employed bees visit the food source and share the information with onlooker bees. Onlooker bees then determine the nectar amount in each food source, based on the fitness value. The search space for given data is re-evaluated for better nectar amounts and values are updated. If no better solution exists, if a maximum number of iterations are reached, or if the training error is not decreasing any further, then the food source is abandoned. Scout bees then restart the random search process for a possible new food source. It is worth noting that the number of employed bees is kept equal to the food source. Common control parameters, such as maximum iterations, colony size, and so on, are evenly distributed between all groups of bees to find the optimal solution, such that the designed IDS could stop malicious activities with better accuracy and low false positives.

The primary contributions of this paper are:

- For the development of a novel attack prediction paradigm, a new intrusion detection system using random neural networks and an artificial bee colony algorithm (RNN-ABC) is proposed.
- The performance of the previously published RNN-IDS is further extended to investigate and compare with the proposed random neural network intrusion detection system (RNN-ABC) using a number of colony sizes, food sources, iterations, and employed bees.
- Results are further extended and error rates are compared on different learning rates for both RNN-IDS and RNN-ABC, using mean of mean squared error (MMSE), standard deviation of mean squared error (SDMSE), best mean squared error (BMSE), and worst mean squared error (WMSE).

The rest of the paper is organized as follows: Preliminary work explaining previous research findings, random neural networks (RNN), and the gradient descent algorithm (GD) is discussed in Section 2. The methodology of RNN-ABC is outlined in Section 3. The results obtained from the adopted methods are explained in Section 4, and the conclusion is presented in Section 5.

2. Preliminary Work

In [13], the authors used a random forest (RF) algorithm for the classification of attack traffic from normal patterns. NSL-KDD was used as a benchmark data set, using 10-fold validation for classification in Waikato Environment for Knowledge Analysis (WEKA). Empirical results showed that the proposed IDS has low false alarm rates and high accuracy for attacks including Denial-of-Service (DoS), Remote-to-Local (R2L), User-to-Root (U2R), and Probe. However, this technique is dependent upon feature reduction for its efficiency, which could result in an unrealistic output.

In [14], the authors presented an overview of restricted-Boltzmann machine, deep neural networks, and recurrent neural network-based deep learning techniques. A fully connected node model (FCN) was introduced and experiments were conducted. The authors concluded that, as compared to the other established machine learning techniques, such as random forest (RF) or support vector machine (SVM), the proposed FCN model produced promising results on the NSL-KDD data set. However, training using hyper-parameter configurations and changes in units may result in higher false positive rates. In [15], the authors proposed a deep learning-based recurrent neural network approach for IDS which was trained using the NSL-KDD data set. The authors compared the technique with many of the traditional classification approaches, such as Random Forest (RF), Naïve Bayesian (NB), and J48, and found that the recurrent neural network-based IDS had high accuracy for both 2-class and 5-class traffic in the NSL-KDD data set. However, the training time was recorded to be high, due to the feedback nature of recurrent neural networks, which can be a major hurdle in the implementation of scalable IDS.

In [16], the authors presented an artificial neural network (ANN)-based IDS and used NSL-KDD as the data set. The system was trained and tested for normal and attack traffic, including DDoS, U2R, R2L, and Probe attacks. The authors mentioned that, in the cases of U2R and R2L, the used data set contained a lesser number of patterns, by default. Levenberg–Marquardt (LM) and quasi-Newton back-propagation algorithms (BFGS) were used for training the proposed IDS. Testing the model confirmed that the proposed IDS had a good anomaly detection rate for 2-class and an accuracy of 81.2%, which was higher than the other reported models. In order to reduce computational times, random neural networks (RNN) [17] have generated significant results on several platforms [18–23]. RNN are easier to execute on hardware [24]. Authors have also compared RNN with ANN and showed that, even though the training time for a feed-forward RNN is longer than that of an ANN, the runtime calculation for RNN was phenomenally less than ANN [25].

In [26], the authors proposed a multi-layer perceptron (MLP)-based hybrid IDS. An artificial bee colony (ABC) algorithm was used to train the neural network (MLP-ABC) for a set of experiments. The authors concluded that accuracy in detecting novel attacks increased with an increase in colony size and number of food sources, but it lacked the attack detection capability for multi-class attacks in the NSL-KDD data set.

In this research, we aim to use swarm intelligence and propose an artificial bee colony (ABC) algorithm-based random neural network intrusion detection system (RNN-ABC). The neurons are trained with ABC and tested against unknown data points. The NSL-KDD data set is used and binary-class detection is performed with adaptive optimization.

2.1. Random Neural Network Model

A novel class of artificial neural networks was proposed by Gelenbe, named random neural network (RNN) [17]. RNNs have been used extensively for pattern recognition [27] and communication, as well as applied in the heating, ventilation, and air conditioning (HVAC) context [20,22] to enhance energy efficiency. However, little research has been reported which has analyzed the effectiveness of RNNs for intrusion detection systems using the NSL-KDD data set.

In the RNN model, neurons trigger excitation and inhabitation states whenever any signal with a positive or negative potential arrives. Upon the reception of a $+1$ signal, a neuron goes into an excitation state, while an -1 input inhibits the previous state of the neuron by the same potential.

Signals travel between N total neurons which are fully connected to each other, as an impulse. The state of a neuron n_i is represented by its potential at time t, and each neuron n_i has state $K_i(t)$ which is a non-negative integer. A neuron n_i is said to be in an excited state if $K_i(t) > 0$, but it is in an idle state if $K_i(t) = 0$. This means that, if the neuron n_i is in an excited state (i.e., $K_i(t) > 0$) it randomly transmits an impulse signal at a rate r_i towards another neuron n_j with the following probabilities:

- It can reach neuron n_j with probability $p^+(i,j)$ as an excitation signal.
- It can reach neuron n_j with probability $p^-(i,j)$ as an inhibitory signal.
- It can depart the neural network with probability $c(i)$.

Mathematically,

$$c(i) + \sum_{j=1}^{N}[p^+(i,j) + p^-(i,j)] = 1, \forall i, \tag{1}$$

$$w^+(i,j) = r_i p^+(i,j) \geq 0; \tag{2}$$

similarly

$$w^-(i,j) = r_i p^-(i,j) \geq 0. \tag{3}$$

Combining Equations (1)–(3)

$$r(i) = (1 - c(i))^-1 \sum_{j=1}^{N}[w^+(i,j) + w^-(i,j)]. \tag{4}$$

In Equation (4), $r(i)$ is the transmission rate between neurons, which can be written as $r(i) = \sum_{j=1}^{N}[w^+(i,j) + w^-(i,j)]$. The "w" matrices determine the weight updates from neurons and always has positive values as it is the product of the signal transmission rate and the probabilities.

In the proposed model, N neurons are connected with each other to share information based on signals, which can be either positive or negative. At neuron (i), if the arrived signal is positive, it is denominated by a Poisson rate $\Lambda(i)$ and a negative signal arrives at Poisson rate $\lambda(i)$. Hence, for each node (i), the output activation function for that neuron is defined as:

$$q(i) = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)}, \tag{5}$$

where

$$\lambda^+(i) = \sum_{j=1}^{n} q(j) r(j) p^+(j,i) + \Lambda(i), \tag{6}$$

and

$$\lambda^-(i) = \sum_{j=1}^{n} q(j) r(j) p^-(j,i) + \lambda(i). \tag{7}$$

The interested reader wishing to further understand the operation of such a network operation is referred to [11,17].

2.2. Gradient Descent Algorithm

In order to obtain the minima of a function, we used a first-order iterative optimization algorithm known as gradient descent (GD). It has been widely accepted as a common training algorithm among researchers. Fundamentally, GD minimizes the cost function. The error cost function can be denoted as:

$$E_p = \frac{1}{2}\sum_{i=1}^{n}\gamma_i(q_i^p - a_i^p)^2, \gamma_i \geq 0, \tag{8}$$

where $\gamma \in (0,1)$ indicates the state of an output neuron i, q_j^p is an actual differential function, and a_j^p is the anticipated output value. As per Equation (8), in order to find the local minima and reduce the value for error cost function, let us consider the relation between two neurons y and z, where the weights $w^+(y,z)$ and $w^-(y,z)$ are updated as follows:

$$w_{y,z}^{+t} = w_{y,z}^{+(t-1)} - \eta\sum_{i=1}^{n}\gamma_i(q_i^p - a_i^p)[\frac{\partial q_i}{\partial w_{y,z}^{+}}]^{t-1} \tag{9}$$

and, similarly,

$$w_{y,z}^{-t} = w_{y,z}^{-(t-1)} - \eta\sum_{i=1}^{n}\gamma_i(q_i^p - a_i^p)[\frac{\partial q_i}{\partial w_{y,z}^{-}}]^{t-1}. \tag{10}$$

The proposed RNN-IDS model is trained using GD. The calculated weights and biases are updated to the neurons as the algorithm computes the error. The interested reader can learn more details about the GD algorithm in [19].

3. Proposed Random Neural Network-Based Artificial Bee Colony Optimization (RNN-ABC)

In this work, the ABC algorithm is used for training the RNN, as is shown in Figure 1. The ABC algorithm calculates the optimized weights of the RNN. The procedure for finding the optimal weights for the RNN using the ABC algorithm is as follows:

- Step I: Initialize a population of solutions r_a, which are randomly generated by scout bees using the equation
$$r_{ab} = r_{min}^b + rand(0,1)(r_{max}^b - r_{min}^b), \tag{11}$$
where r_{ab} denotes the value of a food source, which ranges from $a = 1, ..., PS$, where PS represents the colony (population) size of the bee colony. Each existing solution is a D-dimensional search space of the parameters to be optimized.

- Step II: Assess the existing population using the default function.

- Step III: Start training the random neural network (RNN), where each solution is formulated using input layer neurons I_n, hidden layer neurons H_i, and output layer neurons O_u of the model. The D-dimensional search space is calculated using the expression
$$[(I_n \times H_i) + (H_i \times O_u)] \times 2. \tag{12}$$
During the training phase, as the weight is randomly distributed among neurons, the food position in the entire population is $r_a = [w_{ih}^{+LoL1} w_{ho}^{+L1L2}, w_{ih}^{-LoL1} w_{ho}^{-L1L2}]$, where $i \in I_n$, $h \in H_i$, $o \in O_u$.

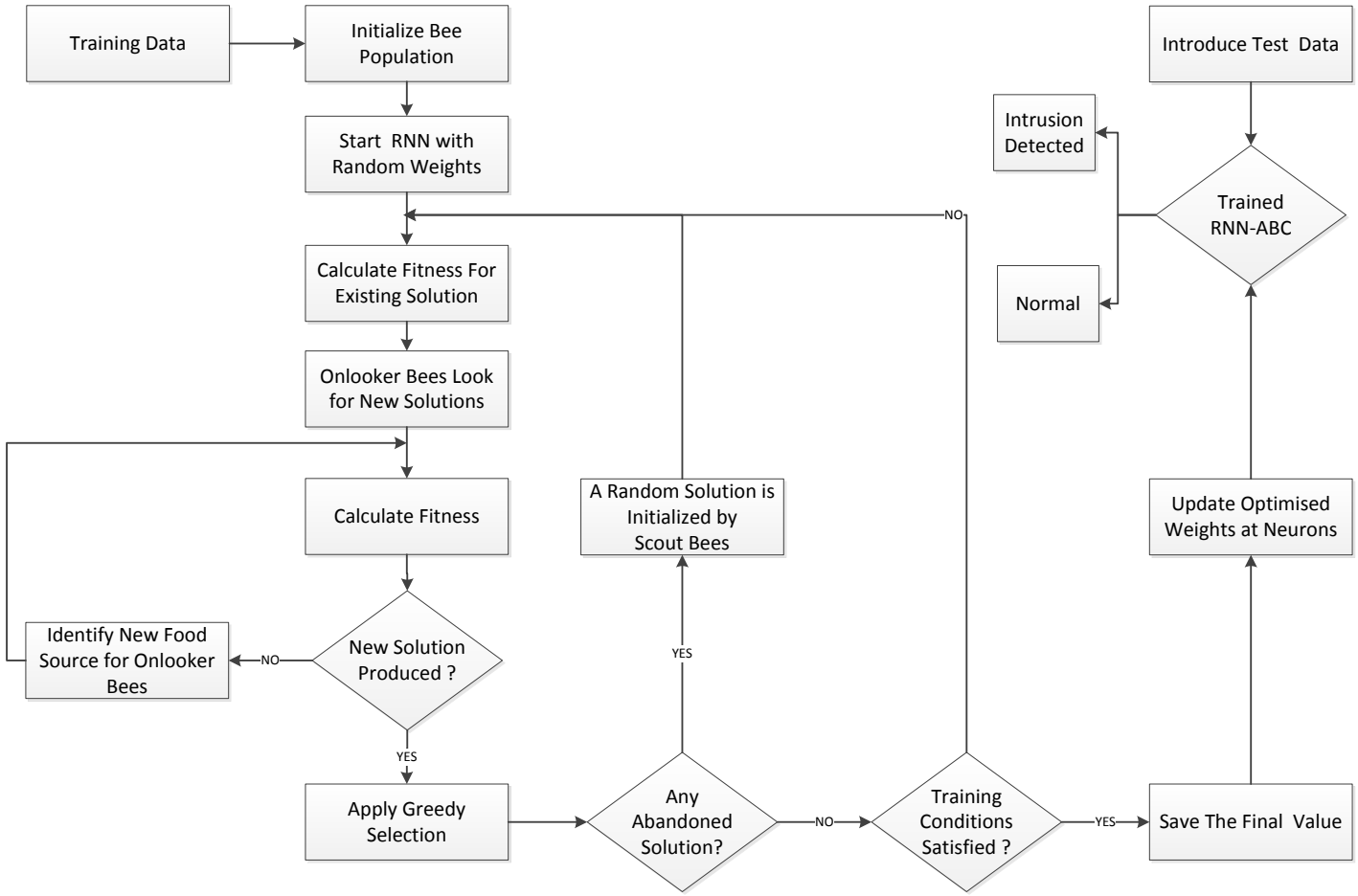


Figure 1. Proposed Architecture For RNN-ABC .

Furthermore,

- w_{ih}^{+LoL1} is a positive potential of mutual weight distribution between neurons i and h of layer 0 and layer 1;
- w_{ho}^{+L1L2} is a positive potential of mutual weight distribution between neurons h and o of layer 1 and layer 2;
- w_{ih}^{-LoL1} is a negative potential of mutual weight distribution between neurons i and h of layer 0 and layer 1; and
- w_{ho}^{-L1L2} is a negative potential of mutual weight distribution between neurons h and o of layer 1 and layer 2.

- Step IV: Evaluation of the fitness (fit_a) of population using objective function:

$$fit_a = \begin{Bmatrix} \frac{1}{1+f(a)} & f(a) \geqslant 0 \\ 1 + |(f(a)| & f(a) < 0 \end{Bmatrix}. \tag{13}$$

- Step V: Calculate the new solution, M_{ab}, as identified by the onlooker bees, and evaluate the fitness of the new source.

$$M_{ab} = r_{ab} + \theta_{ab}(r_{ab} - r_{cb}), \tag{14}$$

where

- $c = 1, 2, ..., PS$;
- $b = 1, 2, ..., D$ are randomly chosen indexes; and
- θ_{ab} is used to control the difference between two neighbour food sources, based on their fitness value.

- Step VI: Start the greedy selection process.
- Step VII: Calculate the probability value P_a of the solution r_a by using Equation (15) and normalize it into the interval [0,1]:

$$P_a = \frac{fit_a}{\sum_{n=1}^{PS} fit_a}. \tag{15}$$

- Step VIII: Onlooker bees find new solution M_{ab} based on the probability P_{ab}.
- Step IX: Calculate the new fitness value fit_a.
- Step X: Re-apply the greedy selection process.
- Step XI: Scout bee to abandon food source if profitability of solution is not improved. New random value is generated using Equation (11).
- Step XII: Store the best solution and erase the previous value, based on fitness.
- Step XIII: Repeat Step IV for a new feasible solution, until maximum iterations or unchanged value of mean square error (MSE) is reached.

3.1. Data Set for Training and Testing

A lot of research related to intrusion detection has been conducted using the DARPA98 and KDDCup99 data sets. Due to the presence of duplicate records, statistical analysis has revealed that attack prediction based upon these data sets is not accurate. In this paper, we used the NSL-KDD data set, which is an abstraction of the previous benchmark KDDCupp99. There are a total of 42 features, in which the first 41 features are used as input and the 42nd feature contains output attack patterns. In the test data set of NSL-KDD, there are a total of 37 different attacks and, in order to create robustness, only 21 of them are used in the training data set. The major attacks consist of Denial-of-Service (DoS), User-to-root (U2R), Root-to-local (R2L), and Probe. A summary of different train and test data sets is illustrated in Table 1.

Table 1. The NSL-KDD data set [14,28].

Dataset Type	Instance	Normal Patterns	Attack Patterns (%)
NSL-KDD Train20	25,192	13,499	46.6
NSL-KDD Train+	125,973	67,343	46.5
NSL-KDD Test+	22,544	9711	56.9
NSL-KDD Test−	11,850	2152	81.8

3.2. Feature Extraction

As discussed above, NSL-KDD contains 41 input features and 1 out-class feature. Machine learning models tend to perform well with features that have fewer co-dependencies and more useful information in them. Having a huge feature space results in biased classification and high false positive rate. In order to increase performance, we aim to reduce the number of input features in this research. Different feature reduction techniques, such as correlation-based feature extraction (CFS), information gain (IG), and gain ratio (GR), have been utilized in [29]. Empirical analysis of the feature space of NSL-KDD revealed that labels such as *dst_host_rerror_rate*, *su_attempted*, and *num_access_files*, along with many others, do not play a significant part in attack detection methods. Hence, in order to reduce the training overhead, we have also reduced the features to 29 (see Figure 2) using the above-mentioned techniques; comparisons are made to check the effectiveness of RNN-ABC.

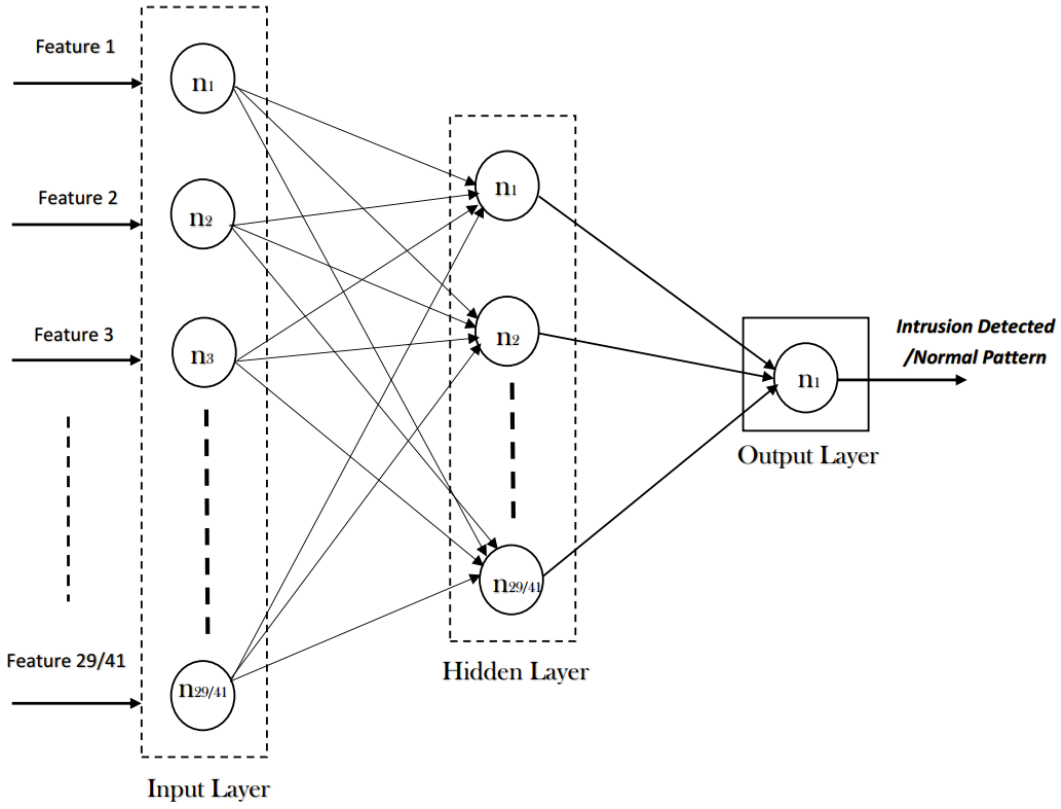


Figure 2. A random neural network model for the design of intrusion detection systems (IDS).

3.3. Data Encoding and Normalization

Most data collected by experiments for intrusion detection are categorical in nature, and may or may not be numeric. Machine learning models are mathematical, which tend to use numerical data. Apart from the few features, such as 'flag', 'protocol-type', and 'service', most of the records are numeric; however, non-numeric values must be converted before using them as an input to RNN-ABC. For this purpose,

'one-hot encoding' was utilized, where a unique binary variable is added against each non-numeric value. This would allow the system to produce better results and avoid unexpected errors.

In the final step, to further reduce the training time of RNN-ABC and to generate unbiased results, the data was further processed using a technique known as normalization. Data is mapped into the [0,1] range, which is most often used where the data points are divergent in nature. We utilized the min-max normalization technique to scale the values, which can be denominated as:

$$s_i = \frac{d_i - \min(d)}{\max(d) - \min(d)}, \tag{16}$$

where $s_{(i)}$ is the normalized output and $d = (d_1, ..., d_n)$ are the input values to normalize.

4. Results

In this section, the following performance metrics are used to report the results in Tables 3–6: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negative (FN) are denoted by α, β, γ, and δ, respectively. Based on the values gathered from these metrics, the accuracy of the IDS is calculated using sensitivity, specificity, false negative rate (FNR), false positive rate (FPR), and positive predictive value (PPV).

4.1. Sensitivity

Sensitivity, also known as the true positive rate (TPR), is defined as the total number of true positives, as identified by RNN-ABC. In other words, it is a true measure of the anomalous data points, which are identified as attacks.

$$Sensitivity = \frac{\alpha}{\alpha + \delta}. \tag{17}$$

4.2. Specificity

Specificity, also known as the true negative rate (TNR), is defined as the total number of true negatives, as identified by RNN-ABC. In other words, it is a true measure of the normal data points which are identified as non-attacks.

$$Specificity = \frac{\beta}{\beta + \gamma}. \tag{18}$$

4.3. False Negative Rate

False negative rate (FNR) indicates the wrong prediction, where RNN-ABC incorrectly predicts an intrusive traffic pattern as a normal pattern.

$$False\ Negative\ Rate = \frac{\delta}{\delta + \alpha}. \tag{19}$$

4.4. False Positive Rate

False positive rate (FPR) is used to report an error in prediction, where RNN-ABC incorrectly predicts a normal traffic pattern as an intrusion.

$$False\ Positive\ Rate = \frac{\gamma}{\gamma + \beta}. \tag{20}$$

4.5. Positive Predictive Value

Positive Predictive Value (PPV), also known as precision, is a measure of the probability of RNN-ABC generating an alert upon detection of anomalous patterns entering the network.

$$Positive\ Predictive\ Value = \frac{\alpha}{\alpha + \gamma}. \tag{21}$$

4.6. Accuracy

Attack prediction of RNN-ABC is considered accurate only if it detects attacks with a low false positive rate and high precision.

Mathematically,

$$Accuracy\,(RNN - ABC) = \frac{\alpha + \beta}{\gamma + \delta + \alpha + \beta}. \tag{22}$$

This paper is an extended version of the publication [11], where a different number of hidden and input layer neurons were used to design the intrusion detection system. The following methods were adopted for training: RNN-IDS with a gradient descent algorithm (GD), and RNN-ABC with an artificial bee colony algorithm (ABC). In addition to this all, methods for both ABC and GD were trained using variant learning rates of 0.4, 0.1, and 0.01, respectively, and additional comparisons were undertaken to discuss the effects of mean square error (MSE) by analysis of mean of MSE (MMSE), standard deviation of MSE (SDMSE), best mean squared error (BMSE), and worst mean squared error (WMSE) [30] during the RNN training and testing phases.

Mathematically:

$$Mean\ Square\ Error = \frac{1}{n}\sum_{i=1}^{n}(\psi_{RNN} - \psi_a)^2, \tag{23}$$

where actual intrusion in the network is denoted as ψ_a and ψ_{RNN} indicates a predicted attack.

4.7. Method-I GD

In the first scenario, reduced features from the NSL-KDD data set, as reported in [16], were used with 29 input layer neurons, 21 hidden layer neurons, and 1 output layer neuron for binary-class classification of attacks.

4.8. Method-II GD

In the second scenario, all 41 features of the NSL-KDD data set were used. As there is no specific formula for selecting the best number of hidden layer neurons, the proposed scheme contained 41 input layer neurons, 41 hidden layer neurons, and 1 output layer neuron for binary-class classification.

4.9. Method-I ABC

As discussed above, more features increase the complexity and training time of the system. For this purpose, we used a reduced feature space of the NSL-KDD, as mentioned in [11,16]. A total of 29 features were extracted from the total feature space of 41. Hence, there were a total of 29 input, 21 hidden, and 1 output layer neurons used. As we utilized swarm intelligence by training the system with the ABC algorithm, the parameters to be optimized were calculated based on Equation (13). The size of the bee colony to carry out the search process was kept 20. A total of 10 employed bees took part in finding the food source during a total of 50 iterations.

4.10. Method-II ABC

In order to validate the performance of RNN-ABC with the results in [11], the complete feature space of NSL-KDD was utilized. The total numbers of input and hidden layer neurons were increased

from the previous method, to 41 input, 41 hidden, and 1 output layer neurons. As mentioned in Table 2, the size of the bee colony remained the same, where 20 employed bees participated in finding the optimal solution over a total of 100 iterations.

Table 2. Training Parameters of RNN-ABC.

Optimization Parameters	Value
Bee Colony Size	20
Food Sources	10
Employed Bees	10
Maximum Iterations	100

After training, the RNN-IDS model was tested against unseen datapoints and results were collected against different performance metrics, as shown in Table 3. The empirical results revealed that, in the case of 21 inputs, the accuracy for the proposed RNN-IDS was 91.4%, which was more than has been reported with artificial neural networks (ANN) in [16]. This suggests that an IDS is good if it is highly sensitive to an intrusion happening in real-time in the network. After trainig the RNN-IDS with the GD algorithm, the empirical results for Method-II suggest that it was 95.60% sensitive to the intrusions, while specificity remained at 69.68%. The system also predicted attacks with a positive predictive value of 98.62%, while the overall accuracy remained at 94.5%.

Table 3. Results for GD RNN-IDS feature-based comparison.

Gradient Decent	Learning Rates					
	Method I			Method II		
	0.4	0.1	0.01	0.4	0.1	0.01
Sensitivity	91.72	93.36	94.24	94.79	95.31	95.60
Specificity	24.8	42.56	55.86	59.62	72.88	69.68
False Negative Rate (FNR)	8.28	5.76	6.64	5.21	5.04	4.40
Flase Positive Rate (FPR)	75.18	57.44	44.14	40.38	27.12	30.22
Positive Predictive Value (PPV)	93.63	96.60	96.95	97.81	98.67	98.62
Accuracy	86.5	91.42	91.02	92.95	94.21	94.50

To check the percentage of error in RNN-IDS trained using the GD algorithm, Table 4 reports the fraction of error by calculating MMSE, SDMSE, BMSE, and WMSE. The performance verified learning rates for both full and extracted features from the NSL-KDD data set in binary classification. For Method I, where there was a reduced number of features, at learning rates of 0.4, 0.1, and 0.01, the MMSE remained at 3.44×10^{-2}, 3.85×10^{-2}, and 3.99×10^{-2}, respectively. Whereas, for Method II, where the complete feature space of the data set was utilized to train the RNN-IDS, the MMSE remained at 3.11×10^{-2}, 3.47×10^{-2}, and 3.9×10^{-2} at disparate learning rates. Empirical analysis backs the results gathered previously in [11], where it was concluded that decreasing the learning rate decreases the error and enhances the efficiency of an intrusion detection system. RNN-IDS with a larger feature space (i.e., method II), outperformed method I, with a BMSE of 3.30×10^{-2} in contrast to 3.72×10^{-2}.

Table 4. Gradient Descent: BMSE, WMSE, SDMSE, and MMSE.

Performance Metrics	Learning Rates					
	Method I			Method II		
	0.4	**0.1**	**0.01**	**0.4**	**0.1**	**0.01**
MMSE	3.44×10^{-2}	3.85×10^{-2}	3.99×10^{-2}	3.11×10^{-2}	3.47×10^{-2}	3.97×10^{-2}
SDMSE	3.42×10^{-2}	3.28×10^{-2}	3.89×10^{-2}	2.73×10^{-2}	2.41×10^{-2}	3.01×10^{-2}
BMSE	5.21×10^{-2}	4.32×10^{-2}	3.72×10^{-2}	4.21×10^{-2}	4.01×10^{-2}	3.30×10^{-2}
WMSE	9.91×10^{-1}	9.47×10^{-2}	9.17×10^{-2}	9.81×10^{-1}	8.44×10^{-1}	7.07×10^{-1}

As an extension of the results from our previous contribution [11], we have developed a new intrusion detection system using a meta-heuristic algorithm inspired by the food search behavior of honey bees, known as an artificial bee colony; the system is called RNN-ABC. Again, two methods have been adopted, with full and reduced feature spaces, respectively, from the NSL-KDD data set. To understand the learning behavior of the RNN-ABC, training was done using rates of 0.4, 0.1, and 0.01, respectively. After testing the trained RNN-ABC with unknown data points, the results in Table 5 reveal that, for Method I, the sensitivity value was reported as 93.98%, with a lowest false negative rate of 6.02% and an accuracy of 93.32%; furthermore, the RNN-ABC method classified unauthorized access with a high precision rate, 97.79%. For Method II, analysis also establishes the fact that increasing the number of neurons at input and hidden layer of the RNN-ABC resulted in better performance, in terms of the sensitivity value of 98.84%. In contrast to Method I, the false positive rate (FPR) decreased from 35.22% to 22.12%. While Method II outperformed Method I in terms of accuracy (95.62%), the fact was once again proved that decreasing the learning rate (0.01) actually increases the accuracy of intrusion detection systems.

Table 5. Results for ABC RNN-ABC feature-based comparison.

Artificial Bee Colony	Learning Rates					
	Method I			Method II		
	0.4	**0.1**	**0.01**	**0.4**	**0.1**	**0.01**
Sensitivity	92.27	93.10	93.98	94.13	95.20	95.84
Specificity	71.15	67.10	64.78	70.70	72.77	77.88
False Negative Rate (FNR)	7.73	6.90	6.02	587	4.80	4.16
False Positive Rate (FPR)	28.85	32.90	35.22	29.30	27.29	22.12
Positive Predictive Value (PPV)	97.64	97.62	97.79	98.21	98.24	99.05
Accuracy	90.75	91.42	92.32	93.10	94.84	95.62

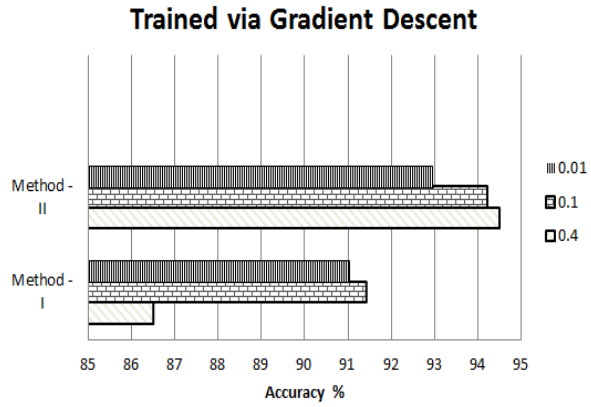
To evaluate the percentage of error in RNN-ABC trained using the ABC algorithm, Table 6 reports the fraction of error by calculating MMSE, SDMSE, BMSE, and WMSE. The MMSE for Method I was 5.21×10^{-2}, 5.89×10^{-2}, and 4.12×10^{-2} for the learning rates of 0.4, 0.1, and 0.01 respectively. For Method II, where there were 42 input and 42 hidden layer neurons, the MMSE was reduced to 3.24×10^{-2}, 3.94×10^{-2}, and 3.87×10^{-2}, respectively, by training the system with the above-mentioned learning rates accordingly. In total, 100 iterations were carried out, where employed bees searched for the optimal solution for the given feature space. Method II surpassed Method I, where the BMSE was 1.92×10^{-2} and SDMSE remained at 2.31×10^{-2}.

It is evident, from the results presented in Figure 3a,b and Tables 3–6, that swarm intelligence-based IDS (e.g., RNN-ABC) trained using an artificial bee colony algorithm takes precedence over an RNN-IDS which is trained using gradient descent algorithm, as in the previous publication [11]. RNN-ABC proved to be more efficient, in terms of a higher true positive rate and a low false positive rate. Additionally, the high sensitivity value in Method-II showed that 95.02% of the time,

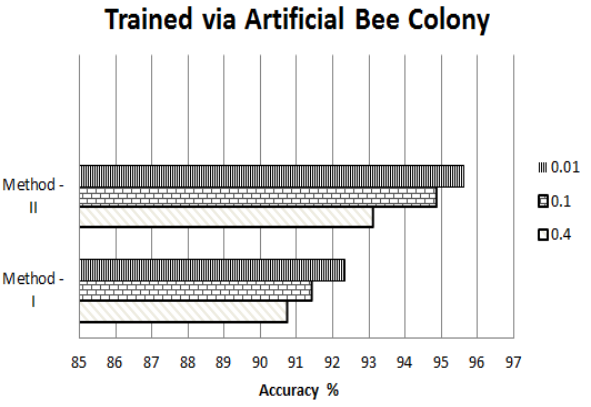
the IDS successfully predicted an attack on the network. Meanwhile, the specificity value of 77.88% shows that RNN-ABC didn't falsely classify normal patterns as an intrusion in the network. It worth noting that there is an inversely proportional relationship between the sensitivity and specificity of a network, where an increase in one value decreases the other. The better performance of RNN-ABC is related to the learning rate, where the value of the error cost function decreases while decreasing the learning rate. This is also due to the size of the bee colony and the number of employed bees, which help to find the optimal solution for a given problem.

Table 6. Artificial Bee Colony: BMSE, WMSE, SDMSE, and MMSE.

Performance Metrics	Learning Rates					
	Method I			Method II		
	0.4	0.1	0.01	0.4	0.1	0.01
MMSE	5.21×10^{-2}	5.89×10^{-2}	4.12×10^{-2}	3.24×10^{-2}	3.94×10^{-2}	3.87×10^{-2}
SDMSE	4.87×10^{-2}	4.74×10^{-2}	2.97×10^{-2}	2.89×10^{-2}	2.76×10^{-2}	2.31×10^{-2}
BMSE	3.12×10^{-2}	3.41×10^{-2}	2.74×10^{-2}	2.01×10^{-2}	2.14×10^{-2}	1.92×10^{-2}
WMSE	9.18×10^{-1}	9.84×10^{-1}	9.79×10^{-1}	9.10×10^{-1}	8.19×10^{-1}	6.20×10^{-1}



(**a**) RNN-IDS



(**b**) RNN-ABC

Figure 3. Accuracy of the RNN with GD and ABC Algorithms.

5. Conclusions

In this paper, we have proposed a novel artificial bee colony algorithm-based random neural network intrusion detection system (RNN-ABC), which uses the advantages of swarm intelligence to detect attacks on a network. This paper is an extension of our previous publication [11], and the same set of methods were adopted to train the neural network with GD and ABC algorithms at several learning rates. Different performance indicators were used to estimate efficiency. The results revealed that the proposed RNN-ABC is more productive than RNN-IDS, as it was highly sensitive to external attacks and the overall efficiency was 95.02%.

It is worth noting that both architectures performed well when all 41 features were used with a learning rate of 0.01. In addition to this, the results were further extended and error rates were compared for both RNN-IDS and RNN-ABC, using mean of MSE (MMSE), standard deviation of MSE (SDMSE), best mean squared error (BMSE), and worst mean squared error (WMSE). The best BMSE for RNN-IDS was 1.92×10^{-2}, which is higher than previously published results for RNN-IDS. Although the training time for ABC was slightly higher than GD, which is more computationally efficient, RNN-ABC performed better overall. The training time can be reduced further by GPU acceleration and manipulating learning rates for corresponding inputs.

In the future, we would like to focus our research in the following ideas:

- To extend our current work for multi-class attack detection with RNN-ABC using other optimization parameters.
- To apply machine learning to design context-aware access-controlled systems which would open the gate for a lot of research opportunities and help to develop and secure several applications, such as IoT, communication systems, and healthcare applications.
- To protect the system from misled prediction, we would like to enhance the capability of RNN-ABC by training it to detect adversarial attacks. This can be achieved by reinforcement mechanisms to improve data reconciliation during the measurement phase of the data set. The performance against adversaries can also be improved using compartmentalization, traffic normalization, applying bifurcating analysis techniques, active mapping, and so on.

Author Contributions: Conceptualization, A.-U.-H.Q. and H.L.; Data curation, J.A. and A.-U.-H.Q.; Formal analysis, A.-U.-H.Q. and A.J.; Methodology, A.-U.-H.Q. and A.J.; Project administration, H.L. and N.M.; Resources, H.L.; Software, N.M., A.-U.-H.Q. and J.A.; Supervision, H.L.; Writing—review & editing, A.-U.-H.Q., H.L. and N.M.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qureshi, A.U.H.; Larijani, H.; Ahmad, J.; Mtetwa, N. A Heuristic Intrusion Detection System for Internet-of-Things (IoT). In Proceedings of the Intelligent Computing—Proceedings of the Computing Conference, London, UK, 16–17 July 2019; Springer: Cham, Switzerland, 2019. [CrossRef]
2. Kayes, A.S.M.; Rahayu, W.; Dillon, T.S. Critical situation management utilizing IoT-based data resources through dynamic contextual role modelling and activation. *Computing* **2019**, *101*, 743–772. [CrossRef]
3. Kayes, A.; Rahayu, W.; Dillon, T.; Chang, E.; Han, J. Context-aware access control with imprecise context characterization for cloud-based data resources. *Future Gener. Comput. Syst.* **2019**, *93*, 237–255. [CrossRef]
4. Iyengar, A.; Kundu, A.; Pallis, G. Healthcare Informatics and Privacy. *IEEE Internet Comput.* **2018**, *22*, 29–31. [CrossRef]
5. Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Process. Mag.* **2018**, *35*, 41–49. [CrossRef]
6. Significant Cyber Incidents. Available online: https://www.csis.org/programs/technology-policy-program/significant-cyber-incidents (accessed on 13 August 2019).

7. Liu, Q.; Li, P.; Zhao, W.; Cai, W.; Yu, S.; Leung, V.C.M. A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View. *IEEE Access* **2018**, *6*, 12103–12117. [CrossRef]
8. Mirjalili, S., Genetic Algorithm. In *Evolutionary Algorithms and Neural Networks: Theory and Applications*; Springer International Publishing: Cham, Switzerland, 2019; pp. 43–55. [CrossRef]
9. Kennedy, J., Swarm Intelligence. In *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*; Zomaya, A.Y., Ed.; Springer US: Boston, MA, USA, 2006; pp. 187–219. [CrossRef]
10. Li, X.; Clerc, M. Swarm Intelligence. In *Handbook of Metaheuristics*; Gendreau, M., Potvin, J.Y., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 353–384. [CrossRef]
11. Qureshi, A.; Larijani, H.; Ahmad, J.; Mtetwa, N. A Novel Random Neural Network Based Approach for Intrusion Detection Systems. In Proceedings of the 10th Computer Science and Electronic Engineering (CEEC), Colchester, UK, 19–21 September 2018; pp. 50–55. [CrossRef]
12. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [CrossRef]
13. Farnaaz, N.; Jabbar, M. Random Forest Modeling for Network Intrusion Detection System. *Procedia Comput. Sci.* **2016**, *89*, 213–217. [CrossRef]
14. Kwon, D.; Kim, H.; Kim, J.; Suh, S.C.; Kim, I.; Kim, K.J. A survey of deep learning-based network anomaly detection. In *Cluster Computing*; Springer: Cham, Switzerland , 2017; pp. 1–13. [CrossRef]
15. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
16. Ingre, B.; Yadav, A. Performance analysis of NSL-KDD dataset using ANN. In Proceedings of the International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96. [CrossRef]
17. Gelenbe, E. *Random Neural Networks with Negative and Positive Signals and Product Form Solution*; MIT Press: Cambridge, MA, USA, doi:10.1162/neco.1989.1.4.502.
18. Emmanuel, R.; Clark, C.; Ahmadinia, A.; Javed, A.; Gibson, D.; Larijani, H. Experimental testing of a random neural network smart controller using a single zone test chamber. *IET Netw.* **2015**, *4*, 350–358. [CrossRef]
19. Ahmad, J.; Larijani, H.; Emmanuel, R.; Mannion, M.; Javed, A.; Phillipson, M. Energy demand prediction through novel random neural network predictor for large non-domestic buildings. In Proceedings of the Annual IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 24–27 April 2017; pp. 1–6. [CrossRef]
20. Javed, A.; Larijani, H.; Ahmadinia, A.; Emmanuel, R.; Mannion, M.; Gibson, D. Design and Implementation of a Cloud Enabled Random Neural Network-Based Decentralized Smart Controller With Intelligent Sensor Nodes for HVAC. *IEEE Internet Things J.* **2017**, *4*, 393–403. [CrossRef]
21. Saeed, A.; Ahmadinia, A.; Javed, A.; Larijani, H. Intelligent Intrusion Detection in Low-Power IoTs. *ACM Trans. Internet Technol.* **2016**, *16*, 1–25. [CrossRef]
22. Javed, A.; Larijani, H.; Ahmadinia, A.; Gibson, D. Smart Random Neural Network Controller for HVAC Using Cloud Computing Technology. *IEEE Trans. Ind. Inform.* **2017**, *13*, 351–360. [CrossRef]
23. Adeel, A.; Larijani, H.; Ahmadinia, A. Random neural network based novel decision making framework for optimized and autonomous power control in LTE uplink system. *Phys. Commun.* **2016**, *19*, 106–117. [CrossRef]
24. Abdelbaki, H.; Gelenbe, E.; EL-Khamy, S. Analog hardware implementation of the random neural network model. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy, 27 July 2000; Volume 4, pp. 197–201. [CrossRef]
25. Mohamed, S.; Rubino, G. A study of real-time packet video quality using random neural networks. *IEEE Trans. Circuits Syst. Video Technol.* **2002**, *12*, 1071–1083. [CrossRef]
26. Mahmod, M.; Alnaish, Z.; Al-Hadi, I.A.A. Hybrid intrusion detection system using artificial bee colony algorithm and multi-layer perceptron. *Int. J. Comput. Sci. Inf. Secur.* **2015**, *13*, 1
27. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arxiv:1409.1556.
28. NSL-KDD—Datasets—Research—Canadian Institute for Cybersecurity. Available online: http://www.unb.ca/cic/datasets/nsl.html (accessed on 3 May 2018).

29. Bajaj, K.; Pradesh, H.; Arora, A.; University, C. Improving the Intrusion Detection using Discriminative Machine Learning Approach and Improve the Time Complexity by Data Mining Feature Selection Methods. *Int. J. Comput. Appl.* **2013**, *76*, 975–8887. [CrossRef]
30. Javed, A.; Larijani, H.; Ahmadinia, A.; Emmanuel, R.; Emmanuel, R. Random neural network learning heuristics. *Probab. Eng. Inf. Sci.* **2017**, *31* , 1–21. [CrossRef]